Chapter 8

Action Systems

In the realm of philosophy, action is defined as the behaviors that agents can perform for a potential or specific purpose in the environment. For example, manipulation, moving, reasoning, and tool utilization can all be considered as fundamental actions that an intelligent agent can execute to fulfill a goal in real-world scenarios. In other words, actions emerge from the goal-oriented engagement of an agent in its environment, reflecting its intent to transform the external world in pursuit of its goals. Therefore, the action system also plays a vital role in differentiating AI agents and foundation models (e.g., LLMs). Generally, existing foundation models have demonstrated impressive performance across various tasks, but their task scope is still limited as they predominantly relies on the original pre-training objective (e.g., next-token prediction). By serving foundation models as brain intelligence, AI agents equipped with action systems can directly engage with their environment and execute complex user intent. Moreover, action systems can support agents to utilize available tools from external environments, thus significantly extending agents' task scopes. Therefore, the design of action systems will also determine the capability of AI agents in perception, decision making, execution, tool utilization, and any other components to align with the human brain. In other words, foundation models lay the groundwork for agents while action systems determine their ultimate potential to achieve complex targets. Designing an effective and comprehensive action system for AI agents is a critical endeavor that involves significant challenges and notable benefits. In Figure 8.1, we demonstrate the execution process of the action system in the cognition system. In this section, we will first discuss the human action system in Section 8.1, and then examine the transition from human action to agentic action in AI agents in Section 8.2. After that, we will systematically summarize the paradigms of existing action systems in AI agents, including action space, action learning, and tool learning, in Section 8.3. In Section 8.4, we analyze the differences between action and perception, and finally we summarize the conclusion in Section 8.5.

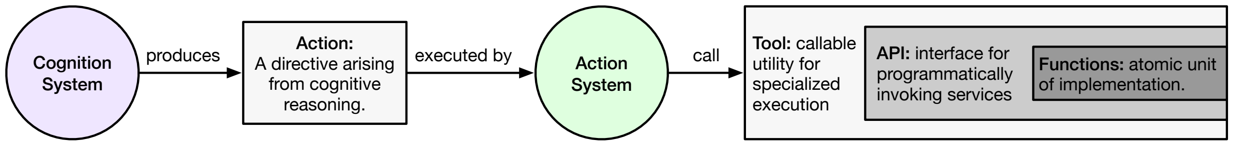

Cognition System
produces
Action: A directive arising from cognitive reasoning.
executed by
Action System
call
Tool: callable utility for specialized execution
API: interface for programmatically invoking services
Functions: atomic unit of implementation.

Figure 8.1: Illustration of several concepts related to action and action execution.

8.1 The Human Action System

Action system in human cognition refers to the processes that allow humans to perceive, plan, and execute goal-directed actions. It is a complex system that enables individuals to interact with a dynamic environment, make decisions, and adapt their behavior based on feedback. Generally, the action system within human cognition could be broadly categorized as *mental action* and *physical action*:

- **Mental action** can be viewed as a kind of distinct action, which is formulated as a thinking process to drive the final intention in the human brain. For example, reasoning, decision making, imagining, and planning can all be considered as various types of mental action. In other words, mental actions are equal to a brain signal that drives the physical actions of humans to fulfill the final objective.

- **Physical action** refers to any goal-directed bodily movement executed by the human motor system. To some extent, physical actions are usually expressed as a kind of continuous action. For example, speaking, manipulating, drawing, running, and grasping can all be regarded as physical actions. Employing a sequence of physical actions, humans can conduct the interaction and collect feedback from real-world environments.

Figure 8.2 illustrates a simple taxonomy of the human action system from the perspective of mental action and physical action. Empowered with both mental and physical actions, the human cognition system can handle diverse complex tasks from real-world scenarios. Drawing inspiration from human cognition, it is also essential for us to revisit how to formulate action systems in AI agents across different tasks, from language to digital and then in physical environments.

Model	Examples	Inputs	Objective	Definition
Large Language Model (LLM)	GPT-4 [7]	Language	Next-Token Prediction	LLM is to generate text based on the provided user prompts.
Large Multimodal Model (LMM)	LLaVA [513]	Multi-modal	Multi-modal Generation	LMM is to generate multimodal data based on multimodal inputs.
Robotic Foundation Model (RFM)	RT-1 [522]	Sensory inputs	Robotic Control	RFM is to generate robotic control based on the sensory inputs from dynamic environments.
Large Action Model (LAM)	LAM [622]	Interactive Environment	Executable Action	LAM is to generate executable actions based on the interactions within the environment.

Table 8.1: Definitions between different kinds of foundation models.

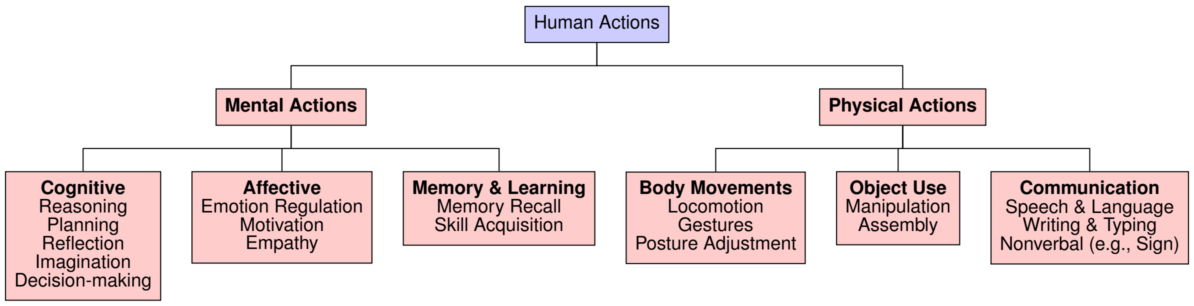


Figure 8.2: Illustrative Taxonomy of Human Actions, showing both mental and physical facets.

8.2 From Human Action to Agentic Action

In the past long period of time, human action systems [623] have significantly motivated us to shape the development of a computer system toward autonomous paradigms. The action mechanism plays a critical role in the human brain in driving goal-directed behavior. In an intelligent human brain [624], conscious and unconscious thinking signals are produced, converted into mental signals, which eventually lead to a sequence of action operations. This process can be mapped as a multi-stage pipeline that involves constructing action spaces, formulating learning mechanisms for improved decision making, and integrating external states (e.g., tools). Inspired by these principles, we discover that these designs are essential to formulate the prototype of AI agent.

Many existing frameworks incorporate action learning into their design or utilize it as an output. To clarify the definition of an action system, we highlight the distinctions among various frameworks, including large language models (LLM), large multi-modal models (LMM), robotic foundation models (RFM), and large action models (LAM), as shown in Table 8.1. Specifically, an LLM is to produce language output based on provided prompts, while an LMM is to generate multi-modality artifacts based on the multi-modal inputs. Existing language-based or digital AI agent frameworks are built upon these foundation models (e.g., LLM or LMM) via predefining the scope of action space and its learning strategies. On the other hand, an RFM is to optimize robotic control based on real-world environments (e.g., robotic video). Existing RFMs are pre-trained from web-scale video data and use video prediction to simulate the action of robotic control. The core of RFM is still to use the generative objective to learn knowledge from large-scale data, although it has involved some action designs in building physical AI agents. Moreover, some recent works [622] introduce the concept of large action model (LAM), which further highlights the stage to generate the action strategies, interact with real-world environments and enhance self-learning paradigm. From these definitions, we notice that, regardless of the foundational models employed, the core of action system is to build the interaction with the environment and then enable the learning process from the collected action trajectories via pre-defined reward functions. Specifically, the mechanisms underlying these behaviors are also similar to the action system in human cognition, offering valuable insights for designing action systems in AI agent frameworks. For example:

- When processing different scenarios, humans usually will pre-define the action space to perform action trajectories to solve specific tasks. For instance, when playing computer games like Minecraft, we will set our action operations via keyboard or mouse to simulate behaviors like building house, mining gold, and so on. On the basis of this, we also need to build or create an action space for handling complex tasks in AI Agent frameworks.
- Compared to machines, the human cognitive system excels in continuously acquiring new knowledge through real-world interactions, guided by generating and optimizing the action sequences. Thus, replicating this learning ability in AI agents is essential to adapt the dynamic environment and build a new skill library.
- In addition, with the development of human civilization, learning to use external tools has been recognized as one of the most significant milestones in the evolution of human intelligence. By leveraging these external tools, humans can extremely extend the problem-solving capability in different scenarios, from the stone age to the industrial revolution.

To this end, we expect to build the mapping between the action system of human cognition system and the design of AI Agent framework, including how to build action space for AI agent from specific scenarios to general domain, how to build action learning within the environment, and how to leverage external states (e.g., tools) to extend the task scope of AI Agent. By developing this a systematic survey, we strive to provide more in-depth insights for the community with a clear understanding of the significance of action systems in AI agent frameworks.

8.3 Paradigms of Agentic Action System

Generally, the action system of AI agent frameworks consists of three major components: 1) the action space $\mathcal{A}$, which includes all types of action that agent can perform in real-world scenarios or downstream tasks, and can vary significantly depending on different agent settings, ranging from language-based agents to embodied agents; 2) the action learning within an dynamic environment that determines the state $\mathcal{S}$, observation $\mathcal{O}$ and the optimization process of agent; 3) the tool space $\mathcal{T}$ that encompasses the instruments, interfaces, or middle-wares the agent can perform for utilization, which ranges from physical devices such as robotic arms to digital interfaces like APIs. Overall, these components collectively define the scope and characteristics of the action system for AI agents, shaping their formulation and execution.

To fully explore the possible actions a_t in practical scenarios, we must formally represent the action space and consider both individual operations and the underlying hierarchical reasoning processes. This means examining the action space at various levels, from low-level manipulations to high-level operators that orchestrate complex workflows.

Accordingly, the AI agent decision making process can be formalized as a trajectory $\langle o_t, s_t, a_t \rangle$, where a_t is selected from the action space $\mathcal{A}$ to transform the current state s_t based on observation o_t into the next state. In some cases, integrating external tool systems may also be necessary. By executing a sequence of $\langle o_t, s_t, a_t \rangle$, the agent is steered toward achieving its final objectives.

8.3.1 Action Space Paradigm

Action space $\mathcal{A}$ is an important component, which serves as the basis for building an action system within AI agent frameworks. The composition of the action space determines how AI agents solve complex tasks in different scenarios. In Figure 8.2, we present an illustrative taxonomy of the action system based on its action space. Generally, we summarize the action space within existing works as three distinct types, as outlined below.

Language Language-based AI agents typically operate through language-driven actions in interactive linguistic environments, such as reasoning, programming, retrieving information, executing API calls, or interacting with external tools. In our study, we summarize three distinct types of language-based action spaces, including plain text, code programming, and communication. Specifically, early language-based AI agents are built with plain text, which aim to perform interactive decision-making in verbal environments or text-based games. Here, ReAct [70] is a representative language-based AI agent, which synergizes the reasoning and actions of an LLM to solve various problems. AutoGPT [625] analyzes and decomposes user requests into multiple subtasks and uses web search or other tools to tackle each of them. Reflexion [48] involves self-refinement and the memory mechanism to enhance action execution in language tasks. LLM+P [163] empowers LLM-based agent with planning capability to aid decision-making. However, converting plain text into an executable command usually requires LLMs to first interpret the text and then perform instruction conversion, leading to additional information loss. To this end, some work explores using

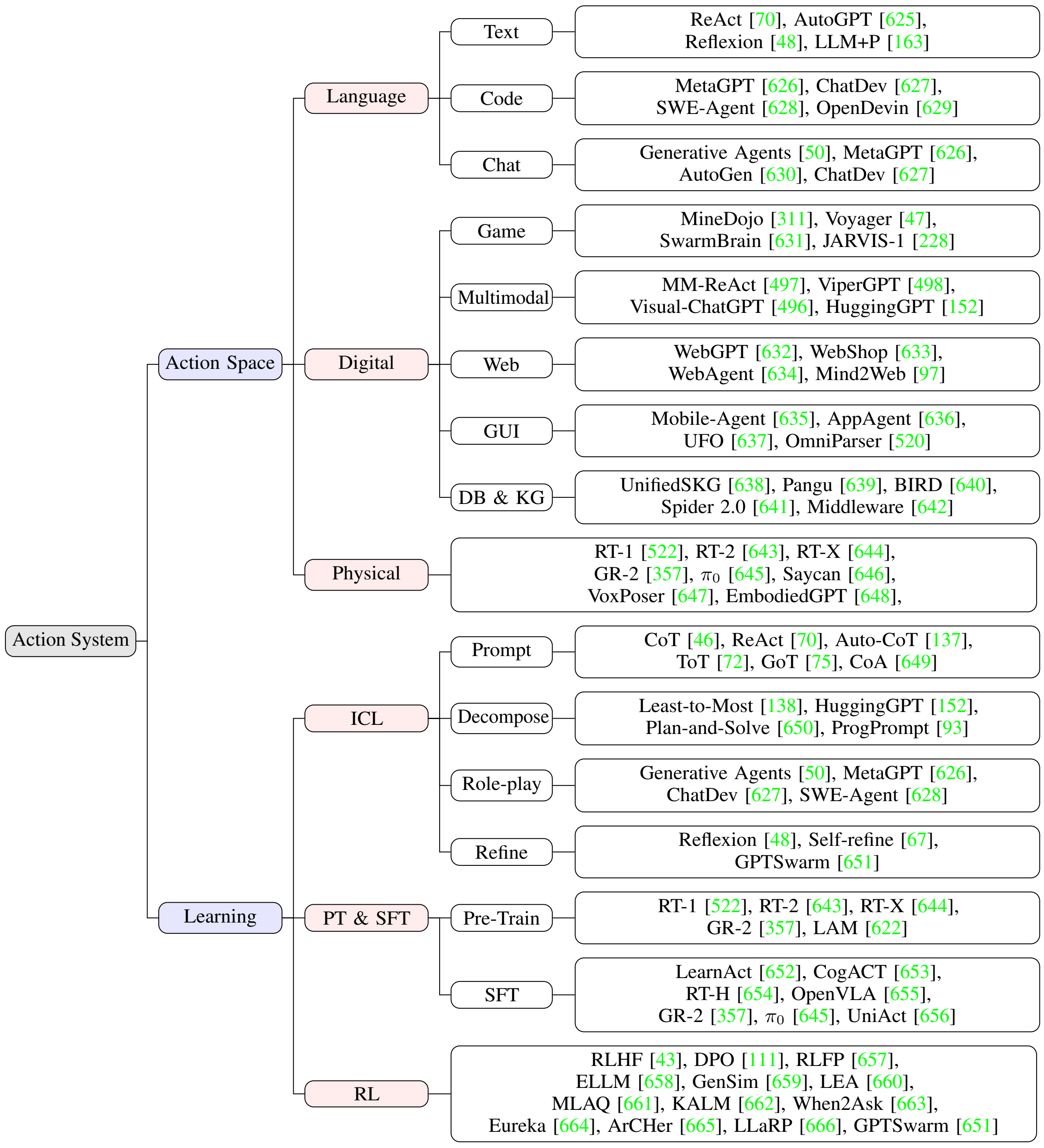

Action System
Action Space
Learning
Language
Digital
Physical
ICL
PT & SFT
RL
Text
Code
Chat
Game
Multimodal
Web
GUI
DB & KG
Prompt
Decompose
Role-play
Refine
Pre-Train
SFT
ReAct [70], AutoGPT [625], Reflexion [48], LLM+P [163]
MetaGPT [626], ChatDev [627], SWE-Agent [628], OpenDevin [629]
Generative Agents [50], MetaGPT [626], AutoGen [630], ChatDev [627]
MineDojo [311], Voyager [47], SwarmBrain [631], JARVIS-1 [228]
MM-ReAct [497], ViperGPT [498], Visual-ChatGPT [496], HuggingGPT [152]
WebGPT [632], WebShop [633], WebAgent [634], Mind2Web [97]
Mobile-Agent [635], AppAgent [636], UFO [637], OmniParser [520]
UnifiedSKG [638], Pangu [639], BIRD [640], Spider 2.0 [641], Middleware [642]
RT-1 [522], RT-2 [643], RT-X [644], GR-2 [357], π_0 [645], Saycan [646], VoxPoser [647], EmbodiedGPT [648],
CoT [46], ReAct [70], Auto-CoT [137], ToT [72], GoT [75], CoA [649]
Least-to-Most [138], HuggingGPT [152], Plan-and-Solve [650], ProgPrompt [93]
Generative Agents [50], MetaGPT [626], ChatDev [627], SWE-Agent [628]
Reflexion [48], Self-refine [67], GPTSwarm [651]
RT-1 [522], RT-2 [643], RT-X [644], GR-2 [357], LAM [622]
LearnAct [652], CogACT [653], RT-H [654], OpenVLA [655], GR-2 [357], π_0 [645], UniAct [656]
RLHF [43], DPO [111], RLFP [657], ELLM [658], GenSim [659], LEA [660], MLAQ [661], KALM [662], When2Ask [663], Eureka [664], ArCHer [665], LLaRP [666], GPTSwarm [651]

Figure 8.3: Illustrative Taxonomy of Action system, including action space and learning paradigm.

code as the action space, allowing direct execution of the generated code and self-verification. MetaGPT [626] and ChatDev [627] build the action space via programming language with multi-agent collaboration. SWE-Agent [628] consider different stages of software engineering and thus solve software issues. OpenDevin [629] devises an automatic software development platform that integrate code writing, interaction with the command, sandbox for code execution, and collaborations. Moreover, some frameworks are built based on multi-agent communications, and then use chatting to analyze which actions should be employed in the next step. Here, Generative Agents [50] directly simulate multiple characters in a virtual town, to explore how each agent to conduct next action. MetaGPT [626] and ChatDev [627] are both multi-agent frameworks to faciliate the development of software engineering. AutoGen [630] is also a representative framework that enable multiple agent collaboration to solve any complex tasks. Generally, language-based AI agents, empowered by LLMs, perform effectively in linguistic interactions. However, limited to the scope of the action space, it also poses challenges of how to solve more complex tasks in real-world scenarios. Therefore, we also need to formulate new research solutions to construct a more sophisticated action space to solve challenging tasks.

Digital To expand the capabilities of AI agents beyond language, some works have also developed advanced AI agents that operate within digital environments, such as web proxies, online shopping platforms, and gaming systems. For examples, MineDojo [311] devises a virtual agent via video-language pre-training and simulates an environment that supports a multitude of tasks and goals within Minecraft. Moreover, Voyager [47] is an embodied AI agent trained to play Minecraft. It simulates multiple executable actions in code form to develop a skill library via interacting with the Minecraft environment, and thus improve the capability of virtual agents. JARVIS-1 [228] is an open-world agent that can handle multi-modal inputs / outputs, generate sophisticated plans, and perform embodied control. It explores the evolutionary behaviors of the agent when acting in Minecraft. SwarmBrain [631] is an embodied agent that uses LLMs to act strategically and in real time in StarCraft II. Additionally, some research studies investigate how LLMs can act to process multimodal tasks. MM-ReAct [497] and ViperGPT [498] apply LLMs to perform the thinking process for multimodal tasks and then select visual experts for task solving. Visual-ChatGPT [496] integrates multiple visual experts and uses LLMs as the controller to solve tasks. HuggingGPT [152] directly involves four stages, including task planning, model selection, model execution and response generation, to automatically analyze user instructions and predict the final answers based on complex multimodal tasks. It is also vital for the agent to keep up with the latest information available online. Therefore, some AI Agent frameworks (e.g., WebGPT [632], WebAgent [634]) are designed to interact with search engine to enhance the capability of agent to discover the answers from website. WebShop [633] is used to explore the potential of AI Agent for online shoping. Mind2Web [97] is to build a generalist agent that simulate multiple complex web tasks. As foundation agents advance in processing multimodal tasks or web tasks, there is a increasing trend to enhance their capability in solving complex computer tasks. Mobile-Agent [635] utilizes multimodal models as the cognitive controller to manage and orchestrate mobile functionalities. AppAgent [636] defines various app usages as action spaces, enabling foundation models to interact with different apps as a mobile intelligent assistant. UFO [637] and OmniParser [520] are two advanced GUI agents which manipulates UI operations as the action space, enabling AI agent to perform computer-use tasks. Generally, empowered with more advanced skills in digital environment, AI agent can demonstrate better intelligent in solving complex tasks, and represent a significant shift from language intelligent to digital intelligent. By expanding the action space to include web browsing, GUI interaction, mobile applications, and embodied systems, AI agents are evolving into more autonomous, multimodal, and context-aware systems, bridging the gap between foundation models and human cognition systems. In addition, other research explores LLM integration with structured digital environments such as relational databases and knowledge graphs (KGs). Pangu [639] pioneered the connection between LLMs and large-scale KGs, while BIRD [640] and Spider 2.0 [641] established a foundation for LLMs to operate with enterprise databases in real-world settings. NL2SQL-BUGs [667] addresses the critical challenge of identifying semantic errors in NL2SQL pipelines [365], which enhances the reliability of LLM-driven interactions with relational databases [668]. Similarly, frameworks like UnifiedSKG [638] and Middleware [642] expand LLMs' action capabilities across both databases and KGs.

Physical Building an AI agent to interact with the real physical world can be viewed as the ultimate objective to simulate a computer program to act as a human cognition system. To achieve this, we require the agent to be capable of processing signals from real-world environments and generating feedback to facilitate continuous improvement. Therefore, it will pose new challenges on how to process the continuous signals collected by sensors and enable foundation models to make decisions. To fulfill this, RT-family [522, 643, 644] pre-trained vision-language-action models to integrate knowledge from web videos into robotic learning, enhancing robotic control and action execution. GR-2 [357] is a robotic model that undergoes large-scale pre-training on video clips and language data, followed by fine-tuning on robot trajectories for robotic action prediction. π_0 [645] pre-trained a robotic model based on robot platforms, including single-arm robots, dual-arm robots, and mobile manipulators, to build robotic learning in physical systems. SayCan [646] bridges the connections between robotic semantics and LLMs, using the robotic model to provide perception for LLMs and then using LLMs to make high-level decision-making. VoxPoser [647] uses LLMs to understand and decompose 3D Value Maps for Robotic Manipulation. Besides, EmbodiedGPT [648] utilizes vision-language models to understand video data and perform decision-driven actions. In physical environments, it is worth noting that we usually need to understand continuous signals and then generate continuous actions for robotic control. Despite the existing foundation models that can effectively process discrete-level actions (e.g., language or computer-use), how to process long continuous signals is still challenging. Therefore, eliminating the differences between continuous signals and discrete signals in foundation models is still a major problem.

Generally, action space serves as one of the most critical components in building an effective AI Agent system. An effective action space enhances the capability and efficiency of the AI Agent in processing downstream tasks. Action space usually ranges from the discrete space (e.g., skill library in Atari games) to the continuous space (e.g., robotic manipulation). As AI agents become more autonomous and multimodal, designing effective action spaces will be crucial for advancing general-purpose AI systems capable of real-world interactions.

8.3.2 Action Learning Paradigm

In the human cognition system, action learning [669] represents the problem-solving process, involving both taking actions and reflecting on feedback. Similarly, action learning for AI agents refers to the iterative process by which an autonomous AI system refines its decision making and behavior through direct interaction with the real world environment. Generally, action learning encompasses a cycle of multiple stages, including building action space, choosing actions, and optimizing action selection based on interaction with the environment (e.g., receiving feedback or rewards and adjusting policy for choosing actions). By iteratively deploying these strategies, AI agents can adapt to the latest information or changing conditions in real time, ultimately enabling more robust, flexible, and efficient problem-solving capabilities. Therefore, an effective action learning mechanism is crucial for the optimization of agentic action systems. In this part, we mainly focus on three different representative learning paradigms, including in-context learning, supervised training, and reinforcement learning, which are discussed below:

In-context Learning As large language models have demonstrated emergent ability, in-context learning has been considered as the most effective method to leverage the existing capabilities of LLM without any modifications. Provided with well-designed prompts to describe actions, AI agents can understand specific actions, perform these actions, reflect on the outcome of the interaction with the environment, and finally achieve goals. Among these approaches, the common method is to use prompting techniques to instruct LLMs to generate agentic action. Here, the most representative one is Chain-of-Thought (CoT) [46] prompting, which applies "*Let us think step by step*" technique to generate a sequence of intermediate reasoning steps, exploring potential solutions systematically. ReAct [70] enables LLMs to generate reasoning trails and task-specific actions through interaction within the environment, improving the reasoning and decision-making capabilities of AI agents. LearnAct [652] devises an iterative learning strategy to expand action space by generating code (i.e., Python) to create and revise new actions. Moreover, some works (e.g., Auto-CoT [137] explores how to automatically generate CoT via LLMs and then enable the autonomous thinking process of AI agents. To handle more complex tasks, ToT [72] considers the thought process as a tree structure and introduces the tree search via LLM prompting, while GoT [75] applies a graph structure along with the graph search. For robotic models, CoA [649] designed four different prompt settings (e.g., object, grasp, spatial, and movement) to allow robot manipulation with reasoning process. Furthermore, to tackle more complex tasks that require intricate agentic workflows, some frameworks introduce the stage of task decomposition via LLM prompting to break down user instructions. Least-to-Most [138] is a classical prompting technique to convert user instructions into multiple subtasks. HuggingGPT [152] is a representative AI agent framework that applies task planning to transform user requirements into actionable items. Plan-and-Solve [650] directly uses LLM to make plans from user instructions and then give answers based on the generated plans. Progprompt [93] applies similar task decomposition to robotic tasks. In addition, using prompting techniques to formulate the characteristic of AI agent has also been considered as an increasing trend to facilitate the simulation and productivity of AI agent frameworks (e.g., Generative Agents [50], MetaGPT [626], ChatDev [627], SWE-Agent [628]). Finally, some other frameworks (e.g., Reflexion [48] or Self-refine [67]) analyze the external feedbacks of user interaction within the environment and then iteratively refine and polish results via well-designed reflexion prompts. All of these designs allow us to better understand user instructions, decompose task goals, and make plans for thinking answers. In-context learning can help us avoid parameter optimization and reduce the heavy cost of training LLMs. It allows AI agents to perform various actions effectively and adapt to a wide range of domains. However, challenges still remain if we want to acquire agents of even stronger action learning ability.

Supervised Training To further improve the action learning ability of foundation models, increasing research efforts have focused on training methodologies, including self-supervised pretraining (PT) and supervised fine-tuning (SFT). For the pre-training paradigm, the most representative works is RT-family [522, 643, 644], which pre-trains robotic Transformer on large-scale web and robotic data, yielding a powerful vision-language-action model. Following this policy, GR-2 [357] is developed through extensive pre-training on a large corpus of web videos to understand the dynamics of the world and post-training on robotic trajectory data to specialize in video generation and action prediction. Similarly, LAM [622] is a large action model pre-trained on trajectories of user interaction with computer usage. However, the pre-training paradigm usually incurs massive computation costs. Therefore, many works take the fine-tuning paradigm to enhance the action capability of foundation models. OpenVLA [670] is built upon the Llama2 [11] language model and incorporates a visual encoder based on DINOv2 [671] and SigLIP [672]. It is fine-tuned on a diverse set of real-world robot demonstrations from Open X-Embodiment (OXE) [673] and outperforms RT-2-X [673] across different tasks, all while utilizing 7$\times$ fewer parameters. Building upon OpenVLA, CogACT [653] integrates an additional diffusion action module and introduces an adaptive action ensemble strategy for inference. It is also fine-tuned using datasets from OXE and demonstrates a 35% improvement in the SIMPLER [674] simulated environment and a 55% increment in real robot tasks using the Franka Arm. Besides, some works also explore how to enable robotic model to learn action from plain language in physical world. For examples, RT-H [654] introduces a hierarchical architecture to build action space, which first predict language motions and then generate low-level actions. And π_0 [645] collected massive diverse datasets from different dexterous robot platforms, and then fine-tune the pre-trained VLMs to learn robotic actions. UniAct [656] learns universal actions that capture generic atomic behaviors across differently shaped robots by learning their shared structural features. This approach achieves cross-domain data utilization and enables cross-embodiment generalizations by eliminating heterogeneity [132]. Overall, using supervised training, including pre-training and supervised fine-tuning, can effectively adapt foundation models to perform actions intelligently in real-world scenarios. Last but not least, it is worth noting that, even with extensive training on a vast corpus, it is still beneficial to apply in-context learning on top of the trained model for AI agents, in an pursuit for their best performance.

Reinforcement Learning To facilitate an action learning procedure in addition to in-context learning and supervised training, it is also crucial for agents to interact with the environment and eventually optimize their action policy through experience, feedback, or rewards. Considering this iterative and sequential nature, reinforcement learning (RL) provides us with the systematic methodology we need [675, 676, 677, 678]. In RL paradigms, there are several classical and representative algorithms, such as Deep Q-Network (DQN) [679] and Proximal Policy Optimization (PPO) [680]. The most representative RL work that applied reinforcement learning to foundation models is InstructGPT [43], which effectively aligns LLM outputs with human preferences via RLHF. Since RLHF usually requires additional training to build the reward model, some papers (e.g. DPO [111]) proposes to directly optimize preference data through contrastive learning. Existing work [89, 681] also demonstrate the potential of scaling the RL algorithm for foundation models to produce long CoT thinking stages with impressive performance. Although RL paradigms have been successfully used to fine-tune LLMs for text generation tasks [12, 682, 43, 683], efficiently utilizing the RL algorithm for action learning remains one of the many challenges that require further attempts. Recent advances indicate significant progress in applying RL to action learning with LLMs from various perspectives:

- Given the rich world knowledge encapsulated in LLM, we can use LLM to *mimic external environments or generate imagined trajectories* to aid agents in action learning. For instance, RLFP [657] utilizes guidance and feedback from the policy, value, and success-reward foundation models to enable agents to explore more efficiently. Similarly, ELLM [658] utilizes large-scale background knowledge from LLMs to guide agents in efficient exploration within various environments. GenSim [659] automatically generates rich simulation environments and expert demonstrations by exploiting the coding abilities of LLM, thereby facilitating the capability of the agent for free exploration. LEA [660] leverages the language understanding capabilities of LLM and adapts LLM as a state transition model and a reward function to improve the performance of offline RL-based recommender systems. MLAQ [661] utilizes an LLM-based world model to generate imaginary interactions and then applies Q-learning [684] to derive optimal policies from this imaginary memory. KALM [662] fine-tunes LLM to perform bidirectional translations between textual goals and rollouts, allowing agents to extract knowledge from LLM in the form of imaginary rollouts through offline RL. In general, empowered by RL paradigms, we can significantly explore the internal knowledge from LLMs and thus enhance the interactions with external environments. Current works such as Search-R1 [685], R1-Searcher [686], RAGEN [687], and OpenManus-RL [688] are exploring utilizing RL methods to fine-tune the agent models on trajectory data in agentic environments.
- Besides, hierarchical RL is also a promising topic that helps foundation model to decompose complex task and then learn optimal policies to solve each task via RL paradigm. For example, When2Ask [663] enables agents to request high-level instructions from LLM. The high-level LLM planner provides a plan of options, and the agent learns the low-level policy based on these options. Eureka [664] leverages LLM to generate human-level reward functions with reflection, allowing agents to efficiently learn complex tasks such as anthropomorphic five-finger manipulation. ArCHer [665] adopts a hierarchical RL approach, utilizing an off-policy RL algorithm to learn high-level value functions, which in turn implicitly guide the low-level policy. LLaRP [666] leverages LLM to comprehend both textual task goals and visual observations. It employs an additional action output module to convert the output of the LLM backbone into a distribution over the action space. Overall, using hierarchical RL can guide AI Agent to explore optimal strategies when analyzing user requests for reasoning and planning.

Using reinforcement learning, we can integrate foundation models with online learning from interactive environments, incorporating both action policies and world models. This integration enables advanced action systems in AI agents. Within the reinforcement learning paradigm, agents dynamically adapt and refine their decision-making processes in

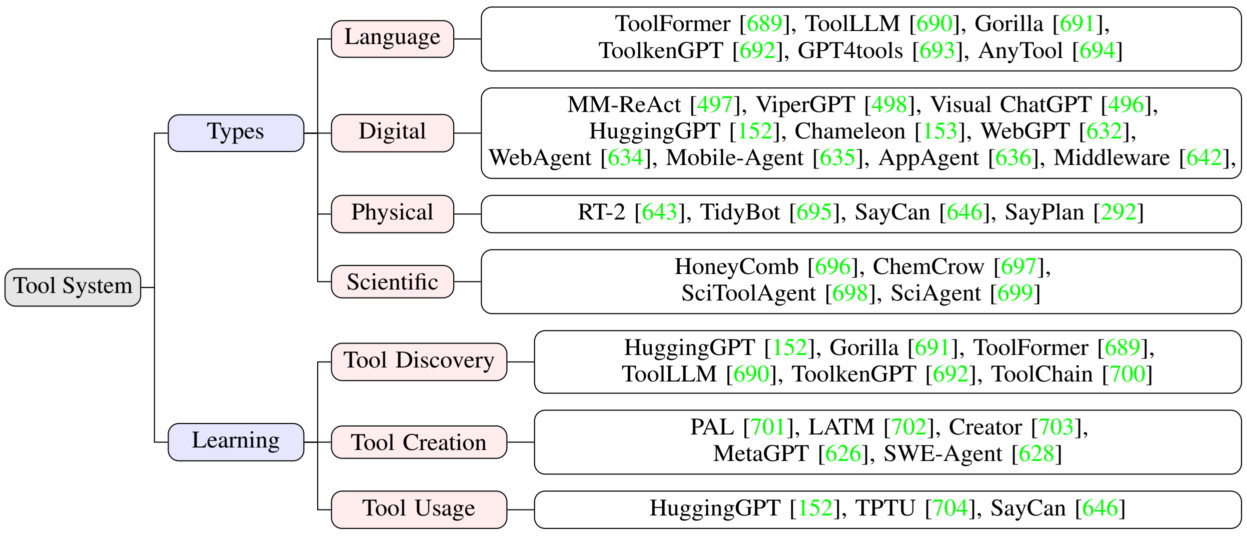

Tool System
Types
Learning
Language
Digital
Physical
Scientific
Tool Discovery
Tool Creation
Tool Usage
ToolFormer [689], ToolLLM [690], Gorilla [691], ToolkenGPT [692], GPT4tools [693], AnyTool [694]
MM-ReAct [497], ViperGPT [498], Visual ChatGPT [496], HuggingGPT [152], Chameleon [153], WebGPT [632], WebAgent [634], Mobile-Agent [635], AppAgent [636], Middleware [642],
RT-2 [643], TidyBot [695], SayCan [646], SayPlan [292]
HoneyComb [696], ChemCrow [697], SciToolAgent [698], SciAgent [699]
HuggingGPT [152], Gorilla [691], ToolFormer [689], ToolLLM [690], ToolkenGPT [692], ToolChain [700]
PAL [701], LATM [702], Creator [703], MetaGPT [626], SWE-Agent [628]
HuggingGPT [152], TPTU [704], SayCan [646]

Figure 8.4: Illustrative Taxonomy of Tool Systems in AI Agents, including tool category and learning paradigm.

response to external feedback, facilitating greater efficiency and effectiveness in action learning and achieving desired outcomes.

Summary In general, Empowered by action systems, AI agents have demonstrated significant decision-making capabilities across various fields. For example, action learning enables AI agents to automate the understanding of Graphical User Interfaces (GUIs) and perform various operations, thereby improving human productivity through automatic computer usage. Moreover, several studies have shown that AI agents equipped with action systems can achieve remarkable outcomes in robotic manipulation tasks, such as object picking, laundry folding, and table cleaning. There are also promising research directions in the industry employing action models. For instance, autonomous driving (AD) has attracted considerable attention due to the exceptional performance of VLMs in perception and decision-making. By integrating human understanding through foundation models, AD systems can effectively comprehend real-world surrounding, enabling them to simulate human-level drivers. In summary, action learning endows agents with the ability to interact with the external world, thereby creating more opportunities for AI applications in real-world scenarios.

8.3.3 Tool-Based Action Paradigm

Tool learning distinguishes human intelligence from that of other animals. Ever since the Stone Age, human use of tools has boosted efficiency, productivity, and innovation. Similarly, enabling AI agents to operate in digital and physical environments by harnessing various tools is a fundamental step toward achieving human-level intelligence.

Definitions In AI, tools are defined as interfaces, instruments, or resources that allow agents to interact with the external world. Examples include web search [632, 705, 97, 634], databases [706, 707, 708, 709], coding environments [710], data systems [711, 712, 713], and weather forecasting [714]. By translating tool functionality into plain text or API formats, foundation models can expand their problem-solving scope. The evolution of tool systems in AI can be summarized in stages. Initially, with the advent of large language models [2], the focus was on converting tools into explainable formats (e.g., function calls). Later, advances in multimodal processing shifted interactions from conversational chats to graphical user interfaces (GUIs), and more recent work has explored embodied agents that control hardware (e.g. robotic arms, sensors) to interact with the physical world. To simplify, a tool-based action can be considered a form of external action employed for assistance.

Tool Category Similar to action spaces, tools can also be classified into multiple categories according to their types. In this part, we mainly summarize three key domains, including language, digital, and physical. In addition, we also explore the potential of tool learning in emerging areas such as scientific discovery:

- *Language:* To facilitate the use of external tools, we usually denote the tool as a kind of function call for foundation models, which usually encompasses task descriptions, tool parameters, and corresponding outputs. This expression allows LLMs to understand when and how to use tools in AI agents. Specifically, ToolFormer [689] expands the capabilities of language models by integrating external tool spaces, including calculator, QA systems, search engine, translation, and calendar. ToolLLM [690] uses RapidAPI as the action space and then uses a depth-first search-based decision tree algorithm to determine the most suitable tool for solving tasks. Gorilla [691] is a fine-tuned LLM based on the tool documents and then can be used to write API calls. ToolkenGPT [692] is to optimize tool embeddings and then enable LLMs to retrieve tools from the fine-tuned tool embeddings. GPT4tools [693] and AnyTool [694] are also building self-instruct datasets and then fine-tune LLMs on them for tool usage. Generally, due to the impressive capability of LLMs, language-based tool utilization for AI agents has been studied, with its effectiveness validated in abundant works, ranging from plain text or function calls to code programming.

- *Digital:* With the success of LLMs in processing language information, many researchers are exploring extending the task scope of AI agents from the language to the digital domains (e.g., MultiModal, Web search, GUI, and so on). For example, MM-ReAct [497], ViperGPT [498], and Visual ChatGPT [496] employed LLMs as the controller and then used LLMs to select visual experts for solving different tasks. HuggingGPT [152] and Chameleon [153] use LLMs to first conduct reasoning and planning actions and then analyze which multimodal tools should be used for solving user instructions. WebGPT [632] and WebAgent [634] respectively empowered LLMs with search engines to enhance the capability of LLMs to solve more challenging tasks. Mobile-Agent [635] and AppAgent [636] respectively incorporate GUI manipulations and App usage as the tool-based actions to extend the task scope of AI agents in solving mobile phone tasks. In contrast to the physical world, digital environments usually provide simpler pipelines to collect and process data. By involving foundation models and their interaction with the digital environment, it is possible for us to develop intelligent assistants in computers, mobile phones, and other digital devices.

- *Physical:* For physical world applications, RT-2 [643] demonstrates language-guided robotic manipulation using visual-language tools, and TidyBot [695] shows how LLMs adapt cleaning tools to personalized household preferences. SayCan [646] uses LLMs as the cognitive system to guide robots in solving tasks through robotic arms and visual perception. SayPlan [292] built a 3D scene graph as the action spaces and designed multiple actions and tools for 3D simulation, and then used LLMs as planners to invoke these actions or tools for robot task planning. Besides, specialized applications in real-world scenarios now also proliferate across different domains. For instance, in surgical robotics, [715] presents a multi-modal LLM framework for robot-assisted blood suction that couples high-level task reasoning, enabling autonomous surgical sub-tasks. Some autonomous driving systems [716, 717] also integrate vision–language models with vehicle control tools for explainable navigation. In total, physical world applications pose the most significant challenge when compared to other tasks, but they also offer the biggest industrial value. Therefore, it still requires us to continue exploring advanced action learning and tool integration in physical-based agents in the future.

- *Scientific:* Scientific tools have played a transformative role in advancing AI agents across disciplines, enabling them to learn, adapt, and execute tasks while integrating foundational models with frameworks that drive innovation and address complex challenges. In materials science, HoneyComb [696] exemplifies tool-driven advancements with its ToolHub. General Tools provide dynamic access to real-time information and the latest publications, effectively bridging gaps in static knowledge bases. Material Science Tools are designed for computationally intensive tasks, leveraging a Python REPL environment to dynamically generate and execute code for precise numerical analysis. Similarly, ChemCrow [697] demonstrates the transformative power of tools in chemistry by integrating GPT-4 with 18 expert-designed tools to automate complex tasks such as organic synthesis, drug discovery, and materials design. These tools include OPSIN for IUPAC-to-structure conversion, calculators for precise numerical computations, and other specialized chemistry software that enables accurate reaction predictions and molecular property evaluations. Similarly, SciToolAgent [698] showcases how multi-tool integration can revolutionize scientific research. Designed to address the limitations of existing systems, SciToolAgent integrates over 500 tools (e.g., Web API, ML models, function calls, databases, and so on). Finally, SciAgent [699] exemplifies a multi-agent framework that integrates ontological knowledge graphs with specialized agents for hypothesis generation and critical analysis, emphasizing the power of modular, tool-driven systems to accelerate discovery in materials science and beyond. These examples underscore the transformative potential of integrating specialized tools into AI frameworks to address domain-specific challenges effectively.

Tool learning Inspired by human evolution [718], the integration of tools in AI involves three key aspects: *Tool Discovery* (identifying suitable tools), *Tool Creation* (developing new tools) and *Tool Usage* (effectively employing tools). We also systematically review existing literature and summarize them in the following:

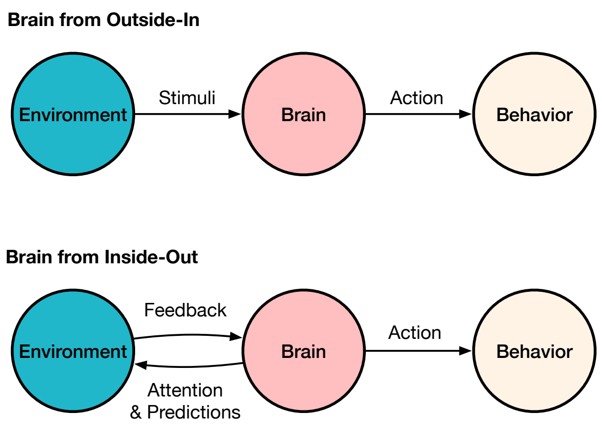
1. **Tool Discovery:** In real-world environments, there is a wide range of tools from the digital to the physical world. Finding the most appropriate tools for user instructions can be challenging. Therefore, the process of tool discovery is to identify and select the appropriate tools that AI agents can operate on to achieve their objectives. This stage also requires the world models in AI agents to have a profound understanding of any complex user instructions and world knowledge of different tools. Moreover, the versatility of AI agents is also correlated with its ability to operate diverse tool systems. Generally, tool discovery can be categorized into two mainstream paradigms: retrieval-based and generative-based methods. Retrieval-based methods aim to select the most relevant tools from the tool library. For example, HuggingGPT [152] introduces a framework in which LLMs act as controllers, orchestrating task planning and then invoking suitable models from platforms such as Hugging Face to fulfill user intention. In generative-based approaches, we often fine-tune LLMs to learn how to use and select tools based on various user instructions. For instance, ToolFormer [689] collects a massive corpus with the corresponding API calls (e.g., calculator, QA system, search engines, translation, and calendar) for training. ToolLLM [690] collect tool instructions based on solution paths and then fine-tune Llama models to generate better API calls for tool utilization.
2. **Tool Creation** In addition to using existing tools, the ability to create new tools plays a crucial role in human civilization. For language agents, a widely adopted approach is to use LLMs to generate functions as executable programs, which consist of both the code and documentation. For example, PAL [701] generates programs as intermediate reasoning steps to solve problems, LATM [702] or Creator [703] use LLMs to create code for user intentions, and to further design a verifier to validate the created tools. SciAgent [699] not only integrates multiple scientific tools but also crafts new tools for scientific discovery. More details on tool creation from an optimization perspective can be found in Section 9.4.2.
3. **Tool Usage** After collecting or creating tools, the effective use of tools constitutes the cornerstone of the capabilities of AI agents, allowing applications that bridge virtual and physical worlds. Modern AI agents increasingly employ tools to tackle complex tasks across diverse domains, with three key dimensions of expansion: 1) *Vertical Specialization*: Agents leverage domain-specific tools to achieve professional-grade performance in complex fields such as robotics, science, and healthcare; 2) *Horizontal Integration*: Systems combine multiple toolkits across modalities (vision, language, control) for multimodal problem-solving; 3) *Embodiment*: Agents physically interact with environments through robotic tools and sensors.

Summary Tool learning and action learning constitute the two most important components of the action system in AI agents. Tool learning can be considered as a kind of action to use external states for problem-solving. Tool learning enables AI agents to substantially broaden their range of tasks, pushing the boundaries beyond the scope of foundation models. For example, empowered by API or function calls, language models can directly reuse the capability of existing models (e.g., retrieval, coding, web search) to generate answers, rather than next-token prediction [719]. Tool learning also involves multiple challenging stages, including how to determine the tool space, how to discover and select tools, and how to create and use tools. Overall, tool learning plays a pivotal role in building an omnipotent AI agent framework to solve complex tasks in different domains.

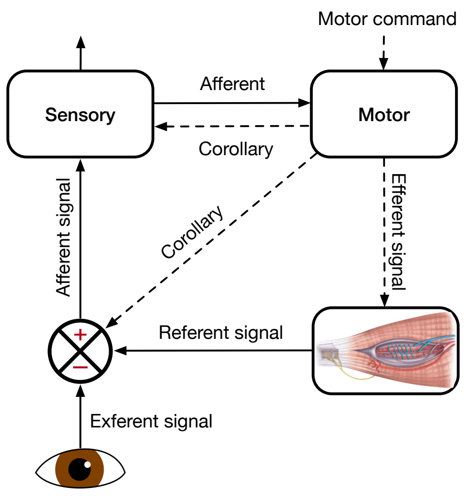
8.4 Action and Perception: "Outside-In" or "Inside-out"

A central debate in cognitive science and neuroscience concerns whether action or perception stands at the root of causal flow in intelligent systems. Figure 8.5 presents different perspectives. The traditional "outside-in" view insists that causal influence begins with external stimuli. The environment excites peripheral receptors, these signals propagate inward, and eventually produce behavior. This perspective portrays the organism—or agent—as essentially reactive: the external world causes sensory changes, and the agent's actions represent a downstream effect of those changes. In contrast, Buzsáki's "inside-out" framework [18] proposes that it is the agent's own actions that shape the meaning and consequences of incoming signals. Such a view implies an active agent, one which continuously generates predictions and motor commands, while sending "corollary discharg" or "action copies" to sensory areas. These internally generated signals serve as references that inform the agent which sensory changes are self-initiated rather than imposed by the outside world. In this manner, cause shifts from an external event to an internally launched initiative, leaving external stimuli to play a confirmatory or corrective role. This reversal has significant implications for how we interpret perception's purpose and function: it is not an end in itself, but a means of updating and refining the agent's own action-driven hypotheses about the environment.

From an evolutionary perspective, possessing the ability to move without relying on sophisticated sensory analysis can yield immediate survival benefits. Even simple organisms profit from periodic motion that stirs up food in nutrient-rich water, long before elaborate perceptual capacities evolve. In other words, movement precedes advanced sensing in evolutionary time, suggesting that the capacity to act is not merely the effect of external stimuli but can



(a) Compare the brain from "Outside-In" and "Inside-Out"



(b) Schematics of the corollary discharge mechanism

Figure 8.5: (a) Compare the brain from "outside-in" and "inside-out". (b) Illustration of the schematic of the corollary discharge mechanism. A motor command (efferent signal) travels from motor areas to the eye muscles, while a corollary discharge (dashed arrow) is routed to a comparator in the sensory system. The comparator uses this internal signal to modulate or subtract external (exafferent) input. Additionally, tension feedback from the muscles (reafferent signal) exerts a delayed effect on perception. Direct projections from motor to sensory cortices underlie this architecture in all mammals. Part (b) is adapted from the original figure in [18].

itself be the driving cause of subsequent perceptual development. It is precisely when action mechanisms become sufficiently established that the agent benefits from additional sensors, which guide those movements more strategically. This developmental sequence grounds perception in utility, tying sensory discrimination to the practical outcomes of movement.

Disruptions in the normal interplay of action and perception illuminate the intricate cause-effect loop. During sleep paralysis, the brain's motor commands temporarily fail to reach the muscles; external stimuli still bombard the senses, but the usual action-to-perception calibration is lost. As a result, the individual experiences a heightened sense of unreality because the brain lacks internally generated reference signals to interpret sensory input. Similarly, if one externally manipulates the eye without the brain issuing a motor command, the visual scene appears to move, highlighting how perception alone—devoid of a preceding, self-initiated action—risks confusion. Neurophysiological data further support the inside-out model. Many neurons in areas once deemed "purely sensory" track not only changes in external stimuli but also self-generated movements—sometimes more strongly so. This indicates that "cause" in the brain frequently emerges from within, guiding both the magnitude and meaning of external signals. Without these internal correlates, raw sensory data can become ambiguous or even useless to the system.

Implications for Intelligent Agents The inside-out perspective offers potent insights for modern research on intelligent agents. Most contemporary AI systems—and many LLM agents—still function predominantly in a reactive mode, awaiting user input and generating responses based on statistical correlations learned from vast datasets. Such passivity resembles an "outside-in" framework, where the agent's role is limited to responding, not initiating. Yet if an agent were to be active, continuously forming and testing hypotheses via self-initiated behaviors (physical or representational), it might ground its own "perceptual" inputs—be they sensory streams or linguistic prompts—and thereby reduce ambiguity. For instance, an LLM-based agent that interjects questions or verifies its own statements against a knowledge base could better discern which inferences are self-caused from those demanded by external data. By tracking these self-initiated contributions (analogous to corollary discharge), the model could improve coherence, lessen errors known as "hallucinations", and refine its internal state through iterative cause-effect loops.

A proactive stance also encourages more data-efficient and context-aware learning. Instead of passively waiting for labeled examples, an agent can explore, provoke feedback, and incorporate self-generated experiences into its training. Over time, this tight coupling between action and perception may bolster the agent's ability to handle complex tasks, adapt to unanticipated challenges, and generalize more robustly. The shift from an outside-in to an inside-out model reframes perception as causally downstream of action. Intelligent systems—whether biological or artificial—stand

Table 8.2: Comparing the perception and action of human and AI agents.

Dimension	Human Brain / Cognition	LLM Agent	Remarks
Perception	- Integrates multiple sensory channels (vision, hearing, smell, touch, taste). - Perception closely tied to emotions, endocrine system, and physical state. - Highly sensitive, capable of detecting subtle differences.	- Primarily language-based with some multimodal capabilities. - Perception depends on external sensors and models with limited integration. - Lacks real-time coupling with physical states.	Perception differences lead to varying ways of understanding reality. Embodied AI attempts to bridge this gap but still faces both hardware and software challenges.
Unified Representation	- Simultaneously processes multimodal inputs: vision, hearing, language, motion, and emotions. - Different brain regions collaborate to create unified spatiotemporal and semantic understanding.	- Primarily text-based. Some multimodal models can process images or audio but with low integration. - No fully unified spatiotemporal modeling like the human brain.	Even advanced multimodal models lack the human brain's holistic, unified representation capacity. Hardware and algorithmic challenges remain.
Granularity in Task Switching	- Flexible in shifting between macro and micro cognitive tasks. - Can plan at a high level and shift focus to finer details when needed. - Adjusts task priority and focus dynamically based on context and working memory.	- Relies heavily on prompt engineering for granularity control. - Cannot autonomously reallocate attention between task layers. - May get stuck in a specific level of abstraction in absence of guided prompts.	Humans can dynamically adjust cognitive granularity based on situational demands, while LLMs require explicit instruction to switch task focus effectively.
Action	- Goal-oriented process drives multiple sensory to make decisions. - Real-time Learning from the experience via the environmental interaction. - Encompass both physical activities and mental processes.	- Action space need to be defined in advance. - Unable to support actions in continuous spaces. - Relies on online training to optimize the decision-making process in the environment.	Humans are capable of actively learning new actions and performing continuous actions, whereas LLM agents currently lack this capability.

to benefit from recognizing that purposeful movement, or proactive conversational steps in the case of LLMs, can actively create, shape, and interpret the signals that flow back in. By acknowledging the cause-effect power of action and striving to build active rather than merely reactive agents, we may approach a deeper understanding of both natural cognition and the next generation of AI.

8.5 Summary and Discussion

Traditionally, action represents the behaviors of the human cognition system based on the interactive feedback from the environment. It endows humans with the capability to think, reason, speak, run, and perform any complex manipulations. Based on the action system, humans can iteratively evolve the brain intelligence by enhancing their perception and actions from the world, and form a closed loop to further create new civilization and innovation in the world. Similarly to a human cognition system, the action system plus the tool system also play an important role for AI agents. Integrating action systems allows AI agents to systematically plan, execute, and adjust their behaviors, facilitating more adaptable and robust performance in dynamic contexts. In this section, we systematically examine and summarize the impact of the action module on AI agents, focusing on both action systems and tool systems.